Versatile Scenario Guidance for Collaborative Virtual Environments

Guillaume Claude, Valérie Gouranton, Bruno Arnaldi

Versatile Scenario Guidance for Collaborative Virtual Environments

Guillaume CLAUDE, Valerie GOURANTON and Bruno ARNALDI
INSA de Rennes, IRISA (UMR6074) - Inria Rennes Bretagne Atlantique, France
{*guillaume.claude, valerie.gouranton, bruno.arnaldi*}*@irisa.fr*

Keywords: Virtual Reality, Scenarios, Collaboration

Abstract: Currently, the scenario of a virtual reality application depends greatly of the representation of the virtual environment and the goals of the application. Different needs arise from these elements such as the abstraction level of their events, the presence of multiple actors or their freedom of action. In this paper, we propose a scenario engine model that aims at being used in any virtual reality application where the scenario is a key feature. It differs from other models as it can be used without making any assumption on the simulation purpose or on the modelling of the Virtual Environment (events model, objects behaviour).

1 INTRODUCTION

In virtual reality, the scenario depicts all the possible sequencing of events allowed in the virtual environment. This description can be more or less rigid depending on its model and the requirements of the simulation. In some cases, the actions are defined step by step. The actor, a character driven by either a user or an autonomous agent system, has limited choices and must follow the scenario step by step. It is generally the case of VR applications dedicated to the training to industrial or surgical procedures. The scenario can also be depicted as a set of rules that describes how the actors have to behave, without much more guidance. This case is widely spread in storytelling where the focus is made more on the characters than on the sequencing of the events. Once defined, the scenario is loaded in a component of the virtual reality system: the scenario engine. The engine uses the scenario to modify the virtual environment and to adapt its inner state to the obtained reactions.

In the context of the development of a virtual reality application framework, we need a model that fits in any simulation where the scenario is a key point. Thus, we do not make any assumption about the purpose of the simulation or its application domain. Furthermore, the events occurring in it are not defined either. The purpose of the simulation also defines the level of guidance provided by the scenario. As an example, if the simulation focuses on the training to a procedure, the scenario may guide the user step by step in his or her actions. It could also wait for the environment to reach specific state to notify the achievement of a goal. Two different scenarios can be applied to a unique virtual environment to provide a version of the simulation with guidance to train the user and a second version without guidance to test his or her training. Finally, the simulation could involve more than one actor. To provide a more accurate model of multi-actor social situations, especially collaboration, we decided to base our work on the role theory (Biddle and Thomas, 1966).

Based on these needs, we have addressed a list of features our ideal scenario model must provide:

- No specific types of events
- No imposed level of guidance
- Role theory modelling for collaboration

2 RELATED WORK

In this section, we present existing work on scenarios for virtual environments. The first two parts are focused on an overview of this work. We then study some of them through the criteria proposed in section 1. In the remainder, we call 'actor' an avatar interacting in the VE and controlled by either a user or an artificial intelligence.

2.1 Emerging Scenarios

Emerging scenario models aim at leaving more freedom to the actors. They define a set of rules, to constraints actors behaviour, that can evolve with time depending on the actors actions and on the state of

Table 1: Synthesis of several Scenario model features

	Events	Guidance	Role
Emergent models	actions of actors	none	Movie like
EMSAVE	"Go forward" notification	static	none
LORA++	STORM only	specified paths	static, describe actions
HAVE	VEHA only	goals to reach	static, actions and goals
StoryNets	Predefined events	static story elements	none

the virtual environment. The sequencing of the events is a consequence, not controlled by the author.

In VRaptor (Shawver, 1997), the scenario emerges from the behaviour of the virtual humans and from the actions (not constrained) of the user. In Facade (Mateas and Stern, 2002), story elements (beats) describe the behaviour of the agents and are designed to respond to the user behaviour in an appropriate manner. Cavazza et al. (Cavazza et al., 2007) proposed a system where the behaviour of the agents is driven by their feelings. The actions of the actors have a set of preconditions and a set of effects on these parameters.

In the remainder, we will not focus more on this kind of scenario engine as we want to focus on the sequencing of the event and on the scenario itself.

2.2 Predefined Scenarios

Predefined scenario engines depict all the possible sequencing of events that may occur in the simulation. Some of them could be limited to a unique execution. In EMSAVE (Vidani and Chittaro, 2009), the actor has several choices during the simulation, but only one is actually able to unfold it. More complex engines describe several possible sequencing of the events that can occur during the simulation. LORA++ (Gerbaud et al., 2007), the StoryNets of MRE (Swartout et al., 2006), or HAVE (Chevaillier et al., 2012) are based on automata and are able to express intricate sequencing of the events.

Predefined scenario engines are generally used for scenarios with a small amount of alternatives. They are reliable when the main concern is to guide the actors through a specific task or story but they generally reach their limits when they are used in a VR system that needs more freedom.

2.3 Modelling of events

To be able to adapt itself to the evolution of the simulation, the scenario engine must be able to perceive the changes occurring in the environment. In many environments, the more common events are the actions of the actors. LORA++ relies on the model STORM (Mollet et al., 2007), able to model collaborative actions. HAVE, a language extended from the UML activity diagram, relies VEHA (Chevaillier et al., 2012) (also an extension of UML) describing the behaviour of the objects in the environment. Other ways to model events can be found in the literature, even if they are not directly connected to scenario models. The work of Willans and Harrison (Willans and Harrison, 2001) proposes to model the behaviour of the objects in the VE using Petri nets. Lugrin and Cavazza (Lugrin and Cavazza, 2007) propose an interaction model based low level physical events sequences recognition.

2.4 Guidance Level

The guidance level of the scenario defines how free is the actor to act at a specific point of the simulation. As stated before, in EMSAVE the actor can only perform only one action. Even if the system offers multiple choices, only one of them will truly affect the virtual environment. Others will simply provide knowledge to the user about why they are not the good option. As they are based on automata, LORA++, HAVE and StoryNets give more freedom to the user. However, LORA++ is specialised in procedural training and the actors are not supposed to do nothing but following the procedure. HAVE proposes an ideal scenario with goals to reach but the actors are free to act as they wish, even not following the scenario. StoryNets offer nodes where the actor is free to act. These nodes are linked together with scripted phases (as in video games cut-scenes). The choice of one link before another depends on what has append during the free phases. However, the user can not intervene in the cut-scenes.

2.5 Role Modelling

The role theory (Biddle and Thomas, 1966) defines the role of a person as what he or she can do, and what he or she has to achieve in regard of a situation and in a specific social context. This definition helps to describe multi-actor organisations such as team work. In virtual reality, this definition fits well to collaborative virtual environments. The "can do" part of the theory defines the actions an actor can execute in regard of the simulation state. The "has to achieve" part

matches to the goals the actors have to reach (alone or collectively).

LORA++ uses roles to define the actions an actor can execute in the virtual environment. This expression of role matches partially with the role theory but lacks dynamic evolution at runtime and goal definition. HAVE defines roles as classes where the possible actions are depicted as well as the goals. These roles are then used as regular UML activity diagram's partitions. The roles are given to users. However, the roles do not seem to evolve during the simulation.

2.6 Synthesis

In the literature, we have not found a scenario model that fits with all of our criteria (see table 1):

- Types of events: Each model is tightly related to another model used to express interaction (LORA++ and STORM) or behaviour (HAVE and VEHA) or to specific kinds of actions (speech recognition for StoryNets or "Go forward" command for EMSAVE).
- Level of guidance: Each of them has a specific level of guidance (from very limited (EMSAVE) to totally free (HAVE)). But none of them is able to adapt itself, making them restricted to specific simulations.
- Role theory: The role theory is not applied or only partially (LORA++ and HAVE)

3 MODEL OVERVIEW

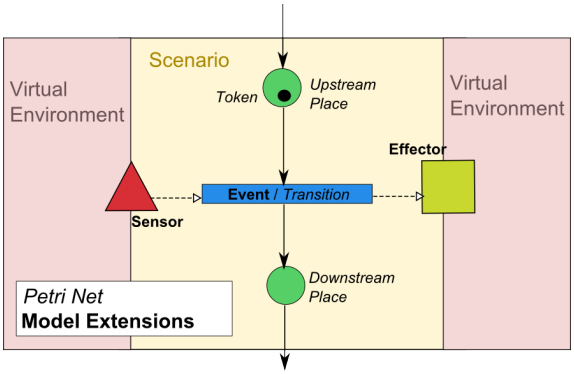

Figure 1: Use of Petri nets to interact with the virtual environment.

In this section, we present our scenario model for virtual reality applications. Its main concerns are:

- Intricate event sequencing,
- Adaptability to any kind of virtual environment regardless of the event model,
- Complete expression of roles,
- Consistency between the parameters in a sequence of events

We aim at providing a solution to problems common to scenarized virtual environments. However, problems raised by technical or applicative contexts are handled by a set of tools used by our model. If needed, new tools can be defined by the user.

3.1 Model Basis: Sequencing and Adaptability

The event sequencing is handled by using safe Petri nets (Murata, 1989) as a base model. We have decided to use Petri nets above state machines because the latter lacks in expressiveness for parallelism. It can be an issue to model partial ordering in the events. Furthermore, Petri nets have been widely studied and extended in the literature. Using Petri nets in Virtual Reality systems is not new. As an example, Smith and Duke (Smith et al., 1999) have applied them to interaction.

The transitions, modelling the events in Petri nets, are completed using two entities, the sensor and the effector (see figure 1). Sensors are the input part of the model. Their job is to wait for specific conditions to be reached. As an example, this condition can be that a specific action has been executed or that a variable in the environment reaches a value. When the condition is met, the transition related to the sensor is triggered. Effectors are the output counterpart. They run a defined behaviour when the related transition is triggered. As an example, a specific type of effectors can be used to modify the roles of the actors in the environment or to start the execution of a script. The sensor and the effector offer to extend the model to be able to adapt it to any kind of event modelling. As an example, a specific type of sensor can be defined to detect low level physical events modelling of actions as proposed by Cavazza et al. (Lugrin and Cavazza, 2007) and an other can be created to match with STORM (Mollet et al., 2007).

3.2 Roles description

In order to fit with role theory, we have embedded in our model the ability to give a fine-grained description of the actions related to roles. Furthermore, these descriptions of roles can be changed dynamically. The role of the actors is expressed by using an element called the attribution. Each transition of the scenario

can be labelled by any number of attributions. Each actor possesses a subset of the attributions used in the scenario. An actor is only able to see the transitions of the scenario labelled by its own set of attributions. Using this feature, the role of an actor is defined by it own set of attributions. More complex conditions can be expressed by this feature. It is possible to allow an actor to view a transition only if he or she has an expected combination of attributions. For example, a transition can only be seen by an actor if he or she has the attribution a and b but not c.

Our model expresses goals by a specific subset of the places, called final places, which needs to hold a token all at once. This set defines a subset of the markings state that, if one of them is reached, the scenario is considered as a success. Another subset of the places, called initial places, defines the initial marking of the scenario. We have also defined that a place can contain a scenario. When a place containing a scenario receives a token, the inner scenario receives a token on all of its initial places. When a final marking is reached, the place is considered as holding a token by the downtream transitions. Using this feature, it becomes possible to express intermediate goals.

3.3 Activity Consistency

As multiple users can have access to the same subnet, one issue left is to maintain the consistency of the parameters in a specific sequence of events. It is achieved by giving transitions the ability to manipulate data stored in the tokens of the Petri net. These data are then used by the sensors when they check that a transition can be triggered or not.

In the remainder of this paper, we call "Activity Continuum" a subnet in which parameters of the events are consistent. For example, if the author of the scenario needs to restrict a sequence of action to the user that started it, the first transition has to store the identity of the actor on the token. The sensors in the continuum then check the identity of the actor and refuse to any other the triggering of a transition. Other use of this feature can be applied to multiple parameters.

3.4 Deadlocks

Deadlocks are a common problem in many systems. In our case they mean that, given the current marking, it does not exist a sequence of events that leads to a final marking. This reachability problem is widely treated, and solved, for Petri nets in the literature (Mayr, 1984). Once the deadlock is detected, solutions can be to prevent the activation of transitions leading to the deadlock or reloading the environment and the scenario to a previous state if a deadlock is reached.

3.5 Comparison with a standard: UML Activity Diagrams

UML Activity Diagrams (UAD) may be seen as sufficient to express Collaborative Virtual Environment scenarios. They are inspired by Petri nets, model concurrency or dispatch the activity between the different stakeholders of a process (Fowler, 2004).

As they are not a scenario model, many of the problems may need a specific solution instead of using generic and reusable components. An example will be the use of a tool. In our model we can express the ability given by a tool to its wielder using assignments. UAD will require to manipulate a specific data structure and to test it each time an action requires this ability. The modelling of several stakeholders is also limited. It will be difficult to distinguish the actors from their roles. Furthermore, UAD do not provide any built-in means to express dynamic changes on the attributions of the roles.

4 MODEL FEATURES

In this section, we detail three features of our scenario engine with different cases:

- Different usage of roles: case of the access to actions
- Expressing constraints on the parameters of a sequence of events: case of the identity of the actors
- Using assignments to model an ability: case of the usage of tools

Figure 2: Our Virtual Reality Application

We have integrated our model in a CVE for the training to a procedure. Our objective is to demonstrate that, with only small changes in the scenario,

we can easily affect the behaviour of the actors and the unfolding of the simulation. These changes could consist to add or remove a transition at a specific point of the scenario, changing the initial attribution of the roles of the actors or adding an activity continuum on a sub-scenario.

4.1 Use Case

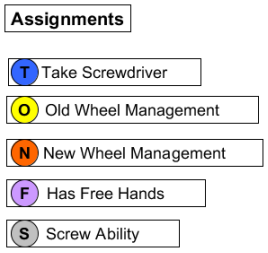

Figure 3: Assignments are a fine grain elements composing roles.

Our VR application aims to train the users to change the wheel of a car. The procedure, and our scenario, is defined by four steps:

- Unscrewing the four screws
- Removing the old wheel
- Installing the new wheel
- Screwing back the screws

To achieve this procedure, we propose eight actions to the actors in our Virtual Environment:

- take/release a screwdriver
- Screw/Unscrew a screw
- Remove/Release the old wheel
- Take/Install the new wheel

Furthermore, the procedure implies some constraints:

- The screws can be removed in any order
- The screws must be screwed crosswise based on the first screw but without constraints on the choice of this screw
- The only way to act upon screws is to possess a screwdriver.

In the following sections we add constraints based on the goals of the simulation. It will impact the scenario without changing the unfolding of the procedure.

We mainly use one type of effector and one type of sensor in our application. The type of the sensors is ”Action sensor”. They aim at triggering the attached transition when a specific action is executed by an actor. The type of the effectors is ”Actor Assignments Modifier”. They add or remove assignments to an actor that triggers the related transition. The roles of the actor is defined using a set of assignments described in Figure 3.

4.2 Role modelling

In our scenario, we have defined three roles: ”Screws manager”, ”Old-wheel manager” and ”New-wheel manager”. In addition, we have used the assignments to express the fact that a user must have the hand free to take an object (the screwdriver or one of the wheels). It is consistent with the role theory as, if an actor has his or her hand free, one can expect him or her to execute specific actions.

Our first use case uses one specific scenario but with two different sets of default roles (see figure 4):

- In the first set, a unique role is assigned to all users. Each user is able to execute any steps of the procedure.
- The second set specifies three different roles (old wheel responsible, new wheel responsible and screwer). Each role is necessary to achieve the scenario.

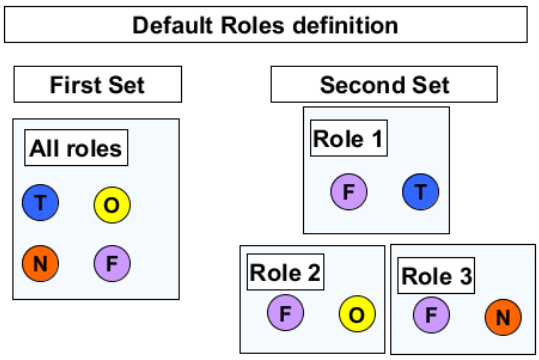

Figure 4: In the first set, each user has the same role. In the second set, specific roles have been created for each task of the procedure.

The second set constrains users to have more specific activities during the simulation. It implies that the simulation must be executed with at least three actors (one for each specific role). The first set allows the scenario to be executed with one actor. Obviously, one can define others sets of roles (e.g. ”screwer” and ”wheels responsible”). Changing this configuration has no impact on the writing of the scenario itself. It is achieved by changing on the initial attributions of the actors. Thanks to our model, we can achieve several different simulations using the same scenario. A video showing an execution of this case can be found at `https://vimeo.com/109443535`

4.3 Activity Continuum

In the real world, stopping an on-going activity and leaving someone else dealing with it is possible most of the time. However, it is generally not the most efficient way to handle things. Usually, the person that starts the activity drives it to its term. It is mainly due to a desire of saving time or energy. While writing a scenario for a CVE, the author may want to express one case or another. In this section we propose the two ways to express a unique scenario.

The first scenario is dedicated to freedom of action. Any actor with the correct assignment combination is able to execute an action. The second scenario, using the activity continuum feature, aims at being closer to the more efficient way to execute the procedure. When an actor execute the first action of the sequence, he or she becomes the only actor able to trigger the downstream transitions up to the last action of the sequence.These two scenarios are opposed in their objectives. Using our model, there is only little changes between the two. In Figure 5 the activity continuum is modelled by the the grey area that links all of the transitions. A video showing an execution of this case can be found at `https://vimeo.com/109446234`

4.4 Tool Usage

In this use case, we focus on how to model the necessity to use a tool in the scenario. We consider the tool as an element giving a specific ability to its owner. In our case, possessing the screwdriver allows an actor to act upon the screws. In this section, we confront two different solutions, among others, to solve this problem. In one of our solution (see Figure 6), the actions to take or put back the tool are integrated as, respectively, first and last steps of the activity. These two actions are fully synchronized with the main part of the activity. Our second solution uses two Petri nets: one is used to express that an actor can take or put back the tool and the other is the activity itself. In this second case (see Figure 7), the actor can take or put back the tool at any time and any actor holding a screwdriver can continue the activity. This solution decorrelates the actions on the tool and the actions on the screws.

The two solutions have a different level of guidance and realism. The first solution restricts more the actions of the actor, focusing on a specific sequencing. The second solution offers more freedom to the actor. He or she can take or release the screwdriver at any moment of the activity. However, without the tool the actor is not able to perform any screwing action. A video showing an execution of this case can be found at `https://vimeo.com/109446235`.

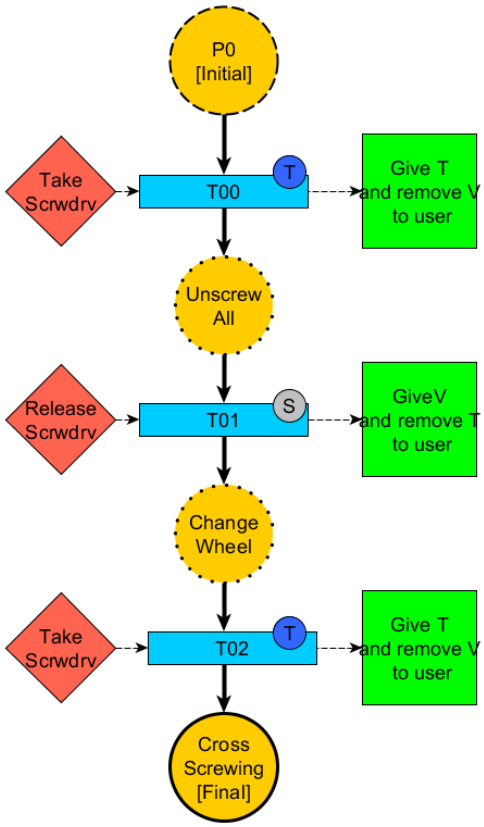

Figure 6: In this scenario, the actor must take the tool just before using it.

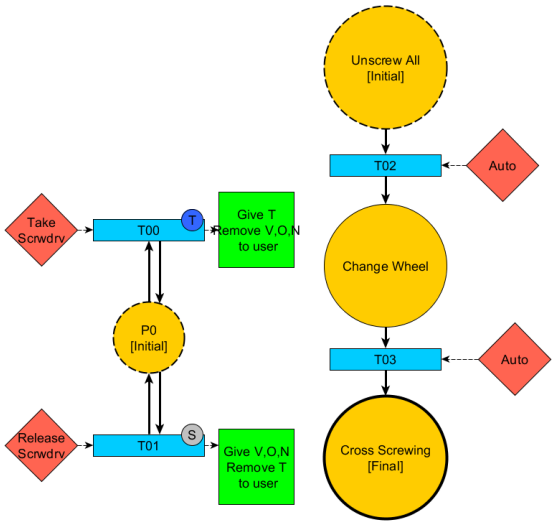

Figure 7: In this scenario the actor must have the right tool to execute specific actions, but there is no constraint on how he has acquired it.

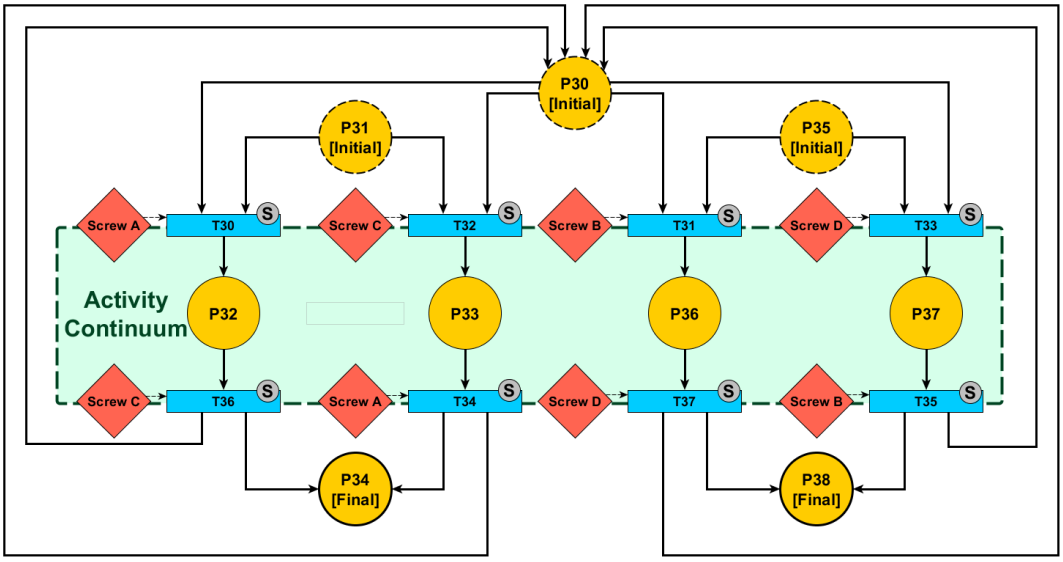

Figure 5: Using the activity continuum (grey area) feature, only the actor that has started the activity is able to finish it.

4.5 Open scenario

Finally, our model can be used to only express constraints on the environment, such as the old wheel must be removed to put the new one, without guiding the actors in the sequencing of their actions. In our previous examples, the scenario guides the actors through the procedure. Here, the scenario defines constraints between the actions and the goal to reach (via the final places). The Figure 8 shows a part of this scenario. This approach is more suited to specific cases such as the evaluation of the users after being trained using one of the previous scenarios. It can also fit to model the interaction constraints in an open world (or sandbox) virtual reality application. We can notice that this scenario is really close to those providing more guidance. In fact, we have reused many of the components.

Combining the two approaches is also possible, thanks to the flexibility of our model. As an example, one can write scenarios with multiple levels of guidance depending on the steps of the simulation.

4.6 Synthesis

Our model allows a scenario to be reused to match changes on criteria such as the purpose of the simulation or the degree of expertise of the a user. Furthermore, It can be achieved thanks to small alterations on the scenario. An other example of the use of our model can be found in (Claude et al., 2014).

5 CONCLUSION AND FUTURE WORK

In this paper we have depicted the features of our scenario engine model:

- Its ability to be connected to any type of events or virtual environments
- The possibility to express intricate scenarios
- The freedom given to the author to define the guidance level
- The expressiveness of the dynamic role modelling based on the role theory
- Its ability to maintain consistency in the parameters of sequences of events

However, our expression of the role theory is not yet complete. Currently, the expression of goals is limited to common goals (intermediate or not). Currently, our model misses the ability to express role related goals. An other main issue left is to detect deadlocks even in parametrized events. Even if the scenario allows to trigger an event, it is possible that the related condition is never be reached in the environment.

AKNOLEDGMENTS

This publication is supported by the S3PM project of the CominLabs Excellence Center. The prototype has been developed with the help of Thomas Boggini, Julian Joseph and Rozenn Bouville-Berthelot.

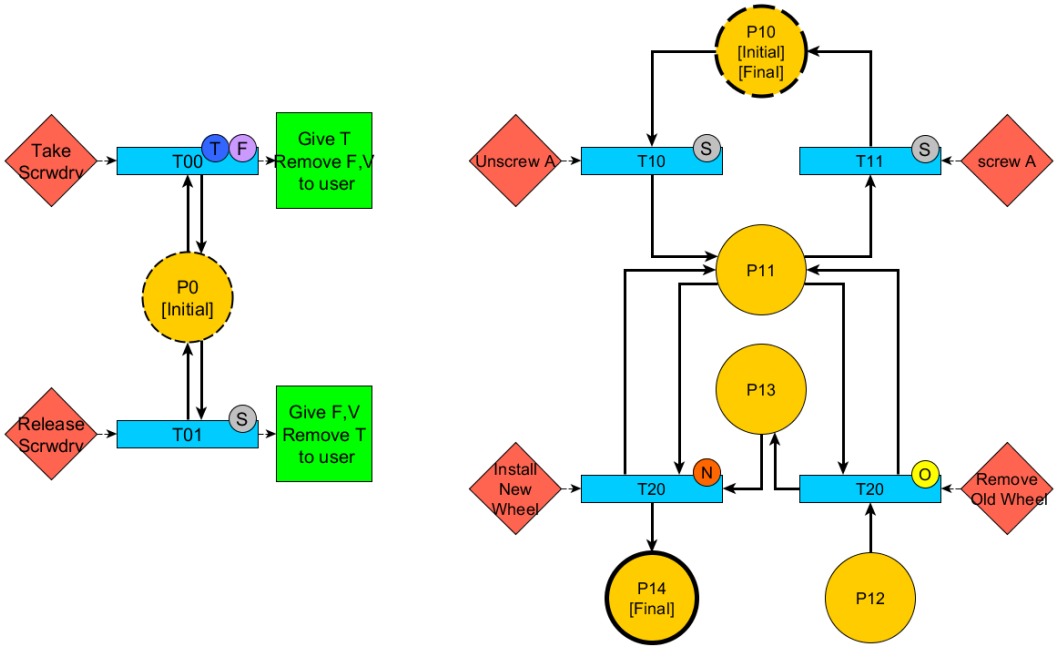

Figure 8: A part of the non-guided version of the scenario.

REFERENCES

Biddle, B. and Thomas, E. (1966). Role theory: concepts and research.

Cavazza, M., Lugrin, J.-L., Pizzi, D., and Charles, F. (2007). Madame bovary on the holodeck: Immersive interactive storytelling. In *Proceedings of the 15th International Conference on Multimedia*, MULTIMEDIA '07. ACM.

Chevaillier, P., Trinh, T.-H., Barange, M., De Loor, P., Devillers, F., Soler, J., and Querrec, R. (2012). Semantic modeling of virtual environments using mascaret. In *Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), 2012 5th Workshop on*. IEEE.

Claude, G., Gouranton, v., Bouville Berthelot, R., and Arnaldi, B. (2014). Short Paper: #SEVEN, a Sensor Effector Based Scenarios Model for Driving Collaborative Virtual Environment. In *ICAT-EGVE*.

Fowler, M. (2004). *UML distilled: a brief guide to the standard object modeling language*. Addison-Wesley Professional.

Gerbaud, S., Mollet, N., and Arnaldi, B. (2007). Virtual environments for training: From individual learning to collaboration with humanoids. In *Technologies for E-Learning and Digital Entertainment*, volume 4469 of *LNCS*. Springer Berlin Heidelberg.

Lugrin, J.-L. and Cavazza, M. (2007). Making sense of virtual environments: action representation, grounding and common sense. In *Proceedings of the 12th international conference on Intelligent user interfaces*. ACM.

Mateas, M. and Stern, A. (2002). A behavior language for story-based believable agents. *IEEE Intelligent Systems*, 17.

Mayr, E. W. (1984). An algorithm for the general petri net reachability problem. *SIAM Journal on computing*, 13(3).

Mollet, N., Gerbaud, S., Arnaldi, B., et al. (2007). Storm: a generic interaction and behavioral model for 3d objects and humanoids in a virtual environment. In *IPT-EGVE the 13th Eurographics Symposium on Virtual Environments*.

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77.

Shawver, D. M. (1997). Virtual actors and avatars in a flexible user-determined-scenario environment. In *Virtual Reality Annual International Symposium*. IEEE Computer Society.

Smith, S., Duke, D., and Massink, M. (1999). The hybrid world of virtual environments. In *Computer Graphics Forum*, volume 18. Wiley Online Library.

Swartout, W., Hill, R., Gratch, J., Johnson, W. L., Kyriakakis, C., LaBore, C., Lindheim, R., Marsella, S., Miraglia, D., and Moore, B. (2006). Toward the holodeck: Integrating graphics, sound, character and story. Technical report, DTIC Document.

Vidani, A. C. and Chittaro, L. (2009). Using a task modeling formalism in the design of serious games for emergency medical procedures. In *Games and Virtual Worlds for Serious Applications, 2009*. IEEE.

Willans, J. and Harrison, M. (2001). Verifying the behaviour of virtual environment world objects. In *Interactive Systems Design, Specification, and Verification*, volume 1946 of *Lecture Notes in Computer Science*.